Final Report: Performance Modeling Activities in PERC2

Summary

Progress in Performance Modeling for PERC2 resulted in:

- Automated modeling tools that are robust, able to characterize large applications running at scale while simultaneously simulating the memory hierarchies of multiple machines in parallel.
- Porting of the requisite tracer tools to multiple platforms.
- Improved performance models by using higher resolution memory models that ever before.
- Adding control-flow and data dependency analysis to the tracers used in performance tools.
- Exploring and developing several new modeling methodologies.
- Using modeling tools to develop performance models for strategic codes.
- Application of modeling methodology to make a large number of "blind" performance predictions on certain mission partner applications, targeting most currently available system architectures.
- Error analysis to correct some systematic biases encountered as part of the large-scale blind prediction exercises.
- Addition of instrumentation capabilities for communication libraries other than MPI.
- Dissemination the tools and modeling methods to several mission partners, including DoD HPCMO and two DARPA HPCS vendors (Cray and IBM), as well as to the wider HPC community via a series of tutorials.

More Robust Modeling Tools

The PERC-2 modeling methodology depends on instruction tracing to acquire *application signatures.* Because the time dilation of a full memory trace can be 100 fold (for instance, using ATOM on Alpha systems) or more (using PIN on Itanium systems, and DyninstAPI on Power4 systems), SDSC has added sampling in time and space to the PERC2 tracing technology (i.e., the MetaSim Tracer). Analysis routines can be toggled on and off in time, and sections of the code can be traced in space (phases). Space sampling allows the parallelization of the tracing process (one can trace different sections of the code in different tracing runs submitted at the same time) and also allows these long running traces to fit into queue runtime limits. With these approaches, ATOM-based tracing runs roughly 30x slower, and PIN-based tracing runs about 100x slower. These slowdowns are an order-of-magnitude faster than previously achieved, thus rendering the study of long-running applications feasible.

Previously (PERC-1) we had to run a separate MetaSim trace run against each memory hierarchy we wanted to simulate. As the number of machines to be modeled has increased (we now model about 26 different processors), and the number of phases required to get full applications traces through queues has increased, this approach became unten-

able. We also stored the memory trace of a full application for subsequent reprocessing; such large datasets became unwieldy. Now the MetaSim Tracer processes the address stream against a cache simulator, in order to predict the performance of that address stream against several different architectures. We have also parallelized the cache simulator, so that we can now do all 26 machines at once, with only minimal additional slowdown (still ~30X for ATOM, and ~100X for PIN). Also, we previously traced just one sample processor of a parallel run, but now we can trace each individual processor's memory access patterns in a parallel run. In addition, we have developed tools, based on SimPoint, for comparing each processor's trace to see how different or similar it is to other processors' traces.

Our current communications tracer is MPIDTrace. We worked with the University of Barcelona (Jesus Labarta) to add tracing for the shmem-like library used by GAMESS and other applications of interest.

As mentioned above, we have ported the MetaSim Tracer to PIN (Itaniums) and DyninstAPI (Power3/4). We are now able to explore how different memory traces are on different machines and compiler options (that is an investigation in progress, partly funded by NSF as well as PERC2).

Improved Modeling Methodologies

We developed a set of metrics and performance models for evaluating the effectiveness of various runtime reordering transformations. The model for iteration reordering is a temporal locality hypergraph. The use of this model involved the development of several new iteration reordering heuristics, as well as the reinterpretation of an existing heuristic. This work was presented at the Second ACM SIGPLAN Workshop on Memory System Performance (MSP 2004).

In an activity that included a subcontract with the University of San Francisco, we applied queuing network models to high performance computer systems. These were used initially to model homogeneous Linux-based clusters. The key issue for this initial effort was how to model various applications stochastically. This extension is of particular interest to USF, as it models the FlashMob system that was assembled there. However, it is also of interest to more traditional HPC centers, which frequently choose to augment existing clusters with additional hardware. In this case, the resulting cluster has limited heterogeneity, which we expect to be fairly simple to capture with queuing network models.

In a separate study, we studied the use of machine learning techniques to automate application performance prediction across large parameter spaces, such as architectural configurations, parallelism or even application input. We are extended our work to the performance prediction of parallel applications.

We also implemented a number of methods for either interpolating or directly measuring the expected memory performance of real loops, namely loops that are not all random or

all stride 1, but some mixture of these as well as other strides. These different options are now built into the PERI Convolver GUI tool. We carried out investigations of which interpolation works best under what circumstances.

We developed a binary analyzer that detects register-carried data dependencies and control flow structures. This is run prior to the trace and then the trace fills in path frequency information and (potentially) memory carried data dependencies (dynamic CFG). We investigated the use of this information to improve the accuracy of models (some loops with same memory signature still run differently due to dependencies).

Validation of Modeling Methodologies on Scientific Applications

We developed performance models of the SCiDAC applications POP (Parallel Ocean Program) and GYRO based on their input parameters. In addition to a simple performance model, we developed a cost model that predicts the optimal performance based only on the hardware parameters. Knowing the optimal execution time of these codes allows us to investigate the compiler effects on performance and serves as a guide and target for performance optimization efforts. In addition, we have developed the design and prototype of modeling assertions for computational regions of an application.

The framework developed by PERC2 has also applied to the workload of our mission partner, DoD HPCMO, to make a set of roughly 170 “blind” performance predictions for TI-05, TI-06, and TI-07. This large number of automatically generated performance predictions was termed ”blind” because PERC2 had access to the machines only via “probes,” i.e. the results of low-level benchmarks. The actual measurement of runtimes was done by a third party (HPCMO staff). These predictions averaged about 10% error. On further analysis some of the “errors” turned out to be poorly run real applications, as for example runs when the system was permitting the application to page. Reconfiguring the machines eliminates this discrepancy. In some other cases the “errors” turned out to be poorly tuned code; better written code should perform closer to the model’s predictions. So in some case it was reality that was wrong not the model! Once the other sources of error were analyzed and corrected (such as those due to dependencies see above) HPCMO adopted the performance modeling framework as official methodology in support of future TI-XX procurements.

Dissemination and Outreach

PERC2’s performance modeling methodology and tools have been presented in a series of tutorials, including at SIGMETRICS and SC05. One of these tutorials, presented at Sigmetrics/Performance 2004, was “Predicting the Performance of Scientific Applications.” Another tutorial was “M10: Methods for Performance Engineering of Scientific Applications,” presented at SC05. Sigmetrics and SC05 tutorials have a very high impact rating, higher than many conferences and journals. For the SC05 tutorial, all PERC2 in-

stitutions participated in one way or another, and reviews were good. Speakers emphasized the various factors that limit performance, including floating point rates, memory bandwidths, and instruction issue rates. Examples were selected from structured computations such as matrix-matrix multiply and unstructured computations such as sparse matrix-vector multiply and mesh-based applications.

PERC2 modeling tools are now in use at DoD HPCMO, Instrumental, and Cray. We are now showing IBM how to use them; they may be used there as well in the future.

Publications

- The PMaC Binary Instrumentation Library for PowerPC
  M. Laurenzano, M. Tikir, L. Carrington, and A. Snavely
  Workshop on Instrumentation and Applications, held in conjunction with ASPLOS XII, October, 2006, San Jose.
- Symbiotic Space-Sharing on SDSC's DataStar System
  J. Weinberg, A. Snavely
  12th Workshop on Job Scheduling Strategies for Parallel Processing In Conjunction with SIGMETRICS 2006, Saint-Malo, France.
- User-guided symbiotic spacesharing of real workloads
  J. Weinberg, A. Snavely
  ICS06 (The 20th ACM International Conference on Supercomputing) , June 2006, Cairns, Australia.
- Path Grammar Guided Trace Compression and Trace Approximation
  X. Gao, A. Snavely, L. Carter
  HPDC-15 (The 15th IEEE International Symposium on High Performance Distributed Computing) , June 2006, Paris.
- ALITER: An Asynchronous Lightweight Instrumentation Tool for Event Recording
  X. Gao, B. Simon, A. Snavely
  Workshop on Binary Instrumentation and Applications (held in conjunction with PACT2005) , September 2005, St. Louis
- Reducing Overheads for Acquiring Dynamic Traces
  X. Gao, M. Laurenzano, B. Simon, A. Snavely
  International Synposium on Workload Characterization (ISWC05), September 2005, Austin
- Low Cost Trace-driven Memory Simulation Using SimPoint
  M. Laurenzano, B. Simon, A. Snavely, M. Gunn
  Workshop on Binary Instrumentation and Applications (held in conjunction with PACT2005) , September 2005, St. Louis.
- Quantifying Locality In The Memory Access Patterns of HPC Applications
  J. Weinberg, M. O. MCracken, A. Snavely, E. Strohmaier
  SC|05 , November 2005, Seattle.
- How well can simple metrics represent the performance of HPC applications?
  L. Carrington, M. Laurenzano, A. Snavely, R. Campbell, L. Davis
  SC|05 , November 2005, Seattle.

- Performance Sensitivity Studies for Strategic Applications
  L. Carrington, X. Gao, N. Wolter, A. Snavely, and R. Campbell
  UGC 2005 , June 2005, Nashville.
- Performance Modeling: Understanding the Present and Predicting the Future
  D.H. Bailey and A. Snavely
  EuroPar 2005 , September 2005, Lisbon.

- Michelle Mills Strout and Paul D. Hovland, "Metrics and Models for Reordering Transformations," in *Proceedings of the Second ACM SIGPLAN Workshop on Memory System Performance (MSP)*, pages 23-34, June 8, 2004.

- Jeff Hollingsworth, Allan Snavely, Simone Sbaraglia, K Ekanadham, "EMPS: An Environment for Memory Performance Studies," *IPDPS 2005, Next Generation Software Workshop*, to appear.